NAT'L INST. OF STAND & TECH
A11106 137708
NIST
PUBLICATIONS
REFERENCE

NISTIR 7148

Relativity of explicit conceptual models

David Flater

QC
100
.U56
#7148
2004

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7148

Relativity of explicit conceptual models

David Flater
Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

July 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arden L. Bement, Jr., Director

Relativity of explicit conceptual models

David Flater

NISTIR 7148
2004-07-29

Abstract

Explicit conceptual models are supposed to capture knowledge of lasting value in a reusable form. Reuse of explicit conceptual models is hampered by arbitrary and application-specific constraints; any constraints that conflict with a new application must be altered or removed before the models can be reused. This paper explores seven facets of relativity in explicit conceptual models using Formal Concept Analysis, demonstrating first that the capture of application-specific constraints is inextricable from the modelling process, and second that the semantic differences between models built for different applications can themselves be modelled formally in most cases. By analyzing those differences, one can determine whether the applications themselves are sufficiently compatible at the conceptual level to enable integration.

Keywords: concept, integration, modelling, ontology, reuse, semantics

1 Introduction

An explicit conceptual model is a conceptual model that has a persistent representation and is intended to capture knowledge of lasting value in a reusable form. Explicit conceptual models are knowledge representations in the broadest sense; they could model the concepts relevant to a software project, a domain of discourse, or any world view of whatever scope.

Explicit conceptual models are sometimes distinguished from implicit conceptual models, which are mental models or intangible models that are implied by the design of an artifact. The latter are not of interest here.

Reuse of explicit conceptual models is hampered by arbitrary and application-specific constraints. Any constraints that conflict with a new application must be altered or removed before the models can be reused. For this reason, the modeller seeks to avoid mingling arbitrary and application-specific constraints with those believed to be universal.

This paper explores seven facets of relativity in explicit conceptual models using Formal Concept Analysis, demonstrating first that the capture of application-specific constraints is inextricable from the modelling process, and second that the semantic differences between

Certain companies, standards, or software systems are mentioned in this paper. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

models built for different applications can themselves be modelled formally in most cases. By analyzing those differences, one can determine whether the applications themselves are sufficiently compatible at the conceptual level to enable integration.

The facets of conceptual relativity to be discussed are shown in Figure 1.

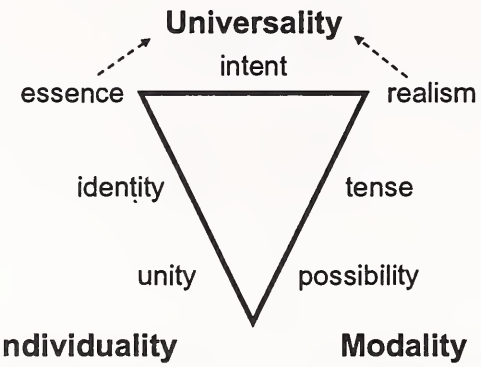

Figure 1: Facets of conceptual relativity

The term *intent* is already well-established in the knowledge modelling domain. The terms *essence*, *identity*, and *unity* were previously introduced to the knowledge modelling community by Nicola Guarino and Christopher Welty [1, 2]. The remaining terms have been introduced in parallel fashion to complete the framework needed to discuss conceptual relativity. The author regrets that philosophical terms may be objectionable to some readers; however, the dilemmas confounding knowledge modellers today were first identified and explored by philosophers, and the related work has no functional equivalent in computer science literature.

As the diagram suggests, the facets are not independent; neither do they fall neatly into categories. However, generally speaking, essence, identity, and unity have to do with individuality—the factoring of the domain of discourse into separate things. Possibility, tense, and realism have to do with modality—the factoring of the domain of discourse into different ways of existing. Essence, realism, and intent have to do with universality—the determination of what is held constant. Essence is an individual perspective on universality (what universality means for individuals); realism is a modal perspective on universality (what universality means for existence).

After notation and terminology are introduced, following sections examine the seven facets of conceptual relativity. These are followed by a discussion of compatibility for integration and the conclusion.

2 Notation

Most of the formal models appearing in this paper are formal contexts and concept lattices as defined in Formal Concept Analysis. The following required definitions are excerpted and condensed from [3, Chapter 1].

> **Definition 1** (was 18). A **formal context** $\mathbb{K} := (G, M, I)$ consists of two sets G and M and a relation I between G and M. The elements of G are called the

objects and the elements of M are called the **attributes** of the context. In order to express that an object g is in a relation I with an attribute m, we write gIm or $(g, m) \in I$ and read it as "the object g **has** the attribute m".

For illustrative purposes a formal context is usually represented as a cross table (see Table 1).

	m
g	$\cdots \quad \cdots \quad \times$

Table 1: Illustration for Definition 1, from [3]

Definition 2 (was 19). For a set $A \subseteq G$ of objects we define

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\}$$

(the set of attributes common to the objects in A). Correspondingly, for a set B of attributes we define

$$B' := \{g \in G \mid gIm \text{ for all } m \in B\}$$

(the set of objects which have all attributes in B).

Definition 3 (was 20). A **formal concept** of the context (G, M, I) is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. We call A the **extent** and B the **intent** of the concept (A, B).

Table 2 illustrates the relationship between the extent (A) and the intent (B) of a concept.

		B	
A		× × × × × × × × × × × × × × ×	

Table 2: Illustration for Definition 3, from [3]

Definition 4 (was 21). If (A_1, B_1) and (A_2, B_2) are concepts of a context, (A_1, B_1) is called a **subconcept** of (A_2, B_2) provided that $A_1 \subseteq A_2$ (which is equivalent to $B_2 \subseteq B_1$). In this case, (A_2, B_2) is a **superconcept** of (A_1, B_1), and we write $(A_1, B_1) \leq (A_2, B_2)$. The relation $\leq$ is called the **hierarchical order** (or simply **order**) of the concepts. The set of all concepts of (G, M, I) ordered in this way... is called the **concept lattice** of the context (G, M, I).

For illustrative purposes a concept lattice is usually drawn as exemplified by Figures 2 *et seq.* Each concept is represented by a circle. The intent of the concept includes all attributes named along upward leading paths from the circle, while the extent of the concept includes all objects named along downward leading paths from the circle.

In most formal contexts, exemplary objects are given the same names as concepts that they exemplify to facilitate intuitive reading of the concept lattice. However, it is formally incorrect to equate any object with a concept that it exemplifies. This issue is discussed in detail in Section 9 (Realism).

Conceptual scaling [3, Section 1.3][4] provides the formal basis for transforming many-valued attributes into one-valued attributes and for reasoning with different granularities (e.g., of space and time).

Definition 5 (was 27). A **many-valued context** (G, M, W, I) consists of sets G, M and W and a ternary relation I between G, M and W (i.e., $I \subseteq G \times M \times W$) for which it holds that

$$(g, m, w) \in I \text{ and } (g, m, v) \in I \quad \text{always imply} \quad w = v.$$

The elements of G are called **objects**, those of M **(many-valued) attributes** and those of W **attribute values.**

Definition 6 (was 28). A **scale** for the attribute m of a many-valued context is a (one-valued) context $\mathbb{S}_m := (G_m, M_m, I_m)$ with $m(G) \subseteq G_m$. The objects of a scale are called **scale values**, the attributes are called **scale attributes.**

Tables 3 and 4 and Figures 2 and 3 demonstrate the standard scales used in this paper. These are only examples; the techniques demonstrated here using the numbers 1 through 4 are applicable to all many-valued attributes.

	1	2	3	4
1	×			
2		×		
3			×	
4				×

Table 3: Nominal scale

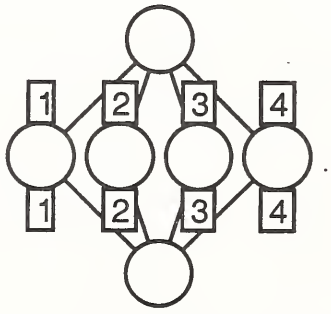

Figure 2: Nominal lattice

	≤ 1	≤ 2	≤ 3	≤ 4	≥ 1	≥ 2	≥ 3	≥ 4
1	×	×	×	×	×			
2		×	×	×	×	×		
3			×	×	×	×	×	
4				×	×	×	×	×

Table 4: Interordinal scale

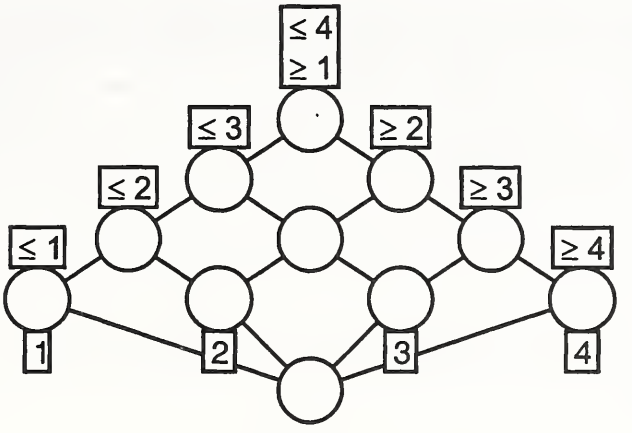

Figure 3: Interordinal lattice

Conceptual time systems with actual objects and time relations are from Temporal Concept Analysis [5, 6, 7, 8], an extension of Formal Concept Analysis. The following definitions from [8, Section 3.1] are required for formal completeness; however, the uses of Temporal Concept Analysis appearing in this paper are sufficiently intuitive that the reader need not be troubled if the definitions are confusing.

Definition 7. Let G be an arbitrary set (of *time objects*) and $T := ((G, M, W, I_T), (S_m \mid m \in M))$ and $C := ((G, E, V, I), (S_e \mid e \in E))$ scaled many-valued contexts (on the same object set G). Then the pair (T, C) is called a *conceptual time system on* G. T is called the *time part* and C the *event part of* (T, C). The scales S_m and S_e of the time and event part describe the chosen granularity structure for the values in these many-valued contexts.

Definition 8. Let P be a set (of 'persons', or 'objects', or 'particles') and G a set (of 'points of time') and $\Pi \subseteq P \times G$ a set (of 'actual objects'). Let (T, C) be a conceptual time system on Π and $R \subseteq \Pi \times \Pi$. Then the tuple (P, G, Π, T, C, R) is called a *conceptual time system (on $\Pi \subseteq P \times G$) with actual objects and a time relation R*, shortly a *CTSOT*.

For each object $p \in P$ the set $p^\Pi := \{g \in G \mid (p, g) \in \Pi\}$ is called the *(eigen)time of p in Π* (which is the intent of p in the formal context (P, G, Π)).

Definition 9. Let (P, G, Π, T, C, R) be a CTSOT, and $p \in P$. Then for any mapping $f : \{p\} \times p^\Pi \to X$ (into some set X) the set $f = \{((p, g), f(p, g)) \mid g \in p^\Pi\}$ is called the *f-life track (or f-trajectory or f-life line) of p*.

For illustrative purposes a life track is usually drawn as an arrow in a concept lattice as exemplified by Figure 7.

3 Modelling terminology

For the purposes of this paper, the Formal Concept Analysis notions on the right of Table 5 are used to model the corresponding generic modelling notions on the left. As a result, the terminologies shown in Table 5 are effectively interchangeable. The semantic differences among modelling environments that are glossed over by this substitution are clarified below.

Generic modelling terminology	Formal Concept Analysis terminology
Class, set, collection, aggregation, type	Concept
Property, attribute	Attribute
Instance, individual, object	Object
Group, composite object	Object or concept

Table 5: Terminology

Process and event related notions are used only indirectly in this paper, so they are not enumerated here. Different views of time, which are central to process models, are discussed in Section 8 (Tense).

Groups / composite objects pose a dilemma for the formal context representation because sometimes they are viewed as whole objects and other times as collections of their parts. When this duality becomes important in Section 6 (Unity), the groups are rendered first as concepts, then as objects using the duality principle for concept lattices.*

The concept lattice in Figure 4 clarifies the semantic differences among terms from Formal Concept Analysis (FCA) [3], Conceptual Graphs (CG) [9], Unified Modeling Language (UML) version 1.5 [10], the Resource Description Framework Schema (RDFS) [11, 12, 13], Web Ontology Language (OWL) [14, 15, 16], the Entity-Relationship Model (E-R) [17], Object Role Modeling [18, 19], EXPRESS [20], Cyc [21, 22], the Suggested Upper Merged Ontology (SUMO) [23, 24], and Frame Logic (F-logic) [25] by characterizing the terms with respect to the five attributes defined in Table 6. Technical notes supporting the characterizations in Figure 4 can be found in the Appendix.

Figure 4 can be used as described in Section 11 (Compatibility for integration) to distinguish the cases in which the substitution of terminology is generally valid with respect to the five modelled attributes from the cases in which additional assumptions are required. For example, an RDFS class can be generalized to an FCA concept, but substituting an FCA concept for a Cyc mathematical set requires the synthesis of an intent (i.e., one must create a new attribute that represents membership in the set).

While the attributes selected for this analysis are sufficient to show coarse similarities and differences, they are not comprehensive. The existence of a subconcept relationship in this formal context does not imply subsumption in the full context of the implicated modelling environment.

* Strictly speaking, the duality is between objects and attributes. In this case, the attribute is one representing membership in the group. The group *per se* is represented by the concept whose intent consists of just that one attribute and whose extent consists of all objects in the group.

A Aggregate (has extent, instances, elements, or members).

C Composite (has parts).

I Has explicit or implicit intent. Generically speaking, an intent is a definition in terms of necessary and sufficient properties from which the extent (instances, elements, or members) can be determined.

P Has attributes / properties. P indicates specifically that the construct has attributes / properties at the model level, not at the meta level. The syntactic attachment of attributes to a class as a way of indicating that every instance of that class has those attributes, as in UML, should not be confused with the notion that the class itself has attributes.

H Has *haecceity* [26]. A haecceity is a transcendental, non-qualitative property that establishes the identity of a thing. If a thing has haecceity, it is always distinct from other things, regardless of how similar they are. Things without haecceity (e.g., mathematical sets) are considered identical if all modelled attributes are the same (e.g., if two sets have the same members).

Table 6: Attributes used in Figure 4

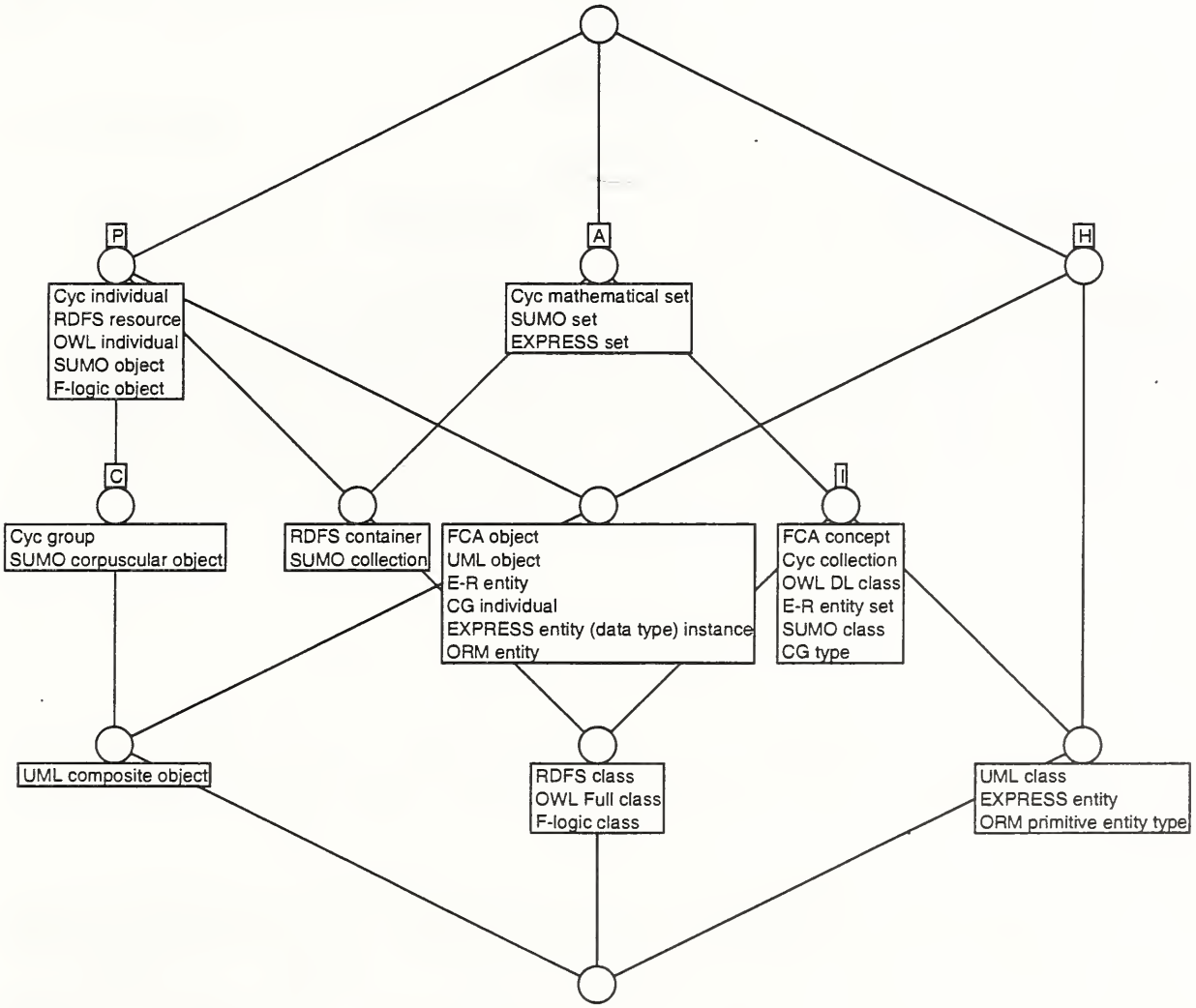

Figure 4: Concept lattice for modelling terms

4 Essence

> A property of an entity is essential to that entity if it must hold for it. This is a stronger notion than one of permanence, that is, a property of an entity is not essential if it just happens to be true of it, accidentally. [1]

To refer to properties as being essential to *individuals* is confusing. As Section 10 (Intent) will emphasize, essence has less to do with the thing itself and more to do with its *classification*. The appropriateness of any given classification is relative to the application.

For example, consider the classification of atoms. A given atom may be classified as a specific isotope, as a specific element, or as just an atom. Having a specific number of neutrons is an essential property of an isotope, but not of an element. Having a specific number of protons is an essential property of an element, but not of an atom.

A concept lattice using helium as an example is shown in Figure 5.

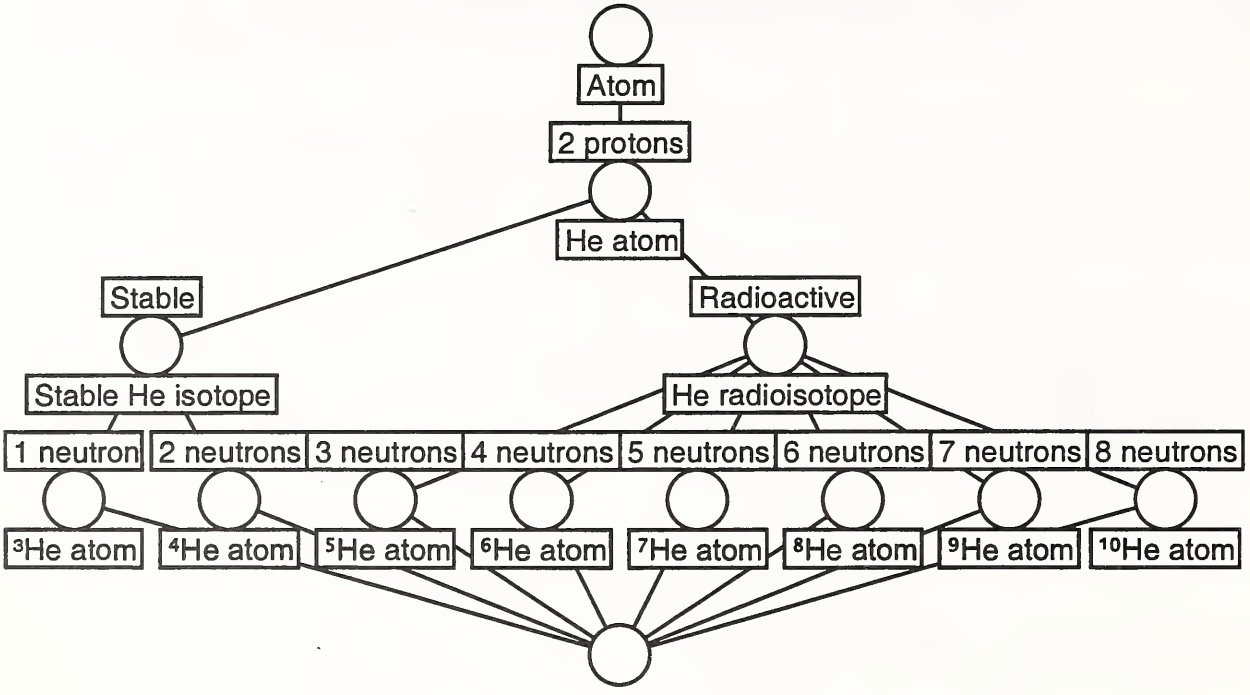

Figure 5: Concept lattice for helium atoms

Suppose that one had a ^{5}He atom that at some point decayed to ^{4}He, emitting a neutron in the process. The ^{5}He effectively ceased to exist, and ^{4}He was created. Yet, from a more general viewpoint, a helium atom existed continuously and merely underwent a non-essential change.

There is nothing special about the selection of properties to be designated essential except in how they relate to the classifications used in an application. Nor is there anything special about the association of terms with classifications. In the above example, it was necessary to use a consistent terminology to make the example understood, but in general, terminology is not used consistently. The term *helium gas* as used by the pilot of an airship would refer only to a naturally occurring mixture in which ^{4}He predominates, while the term *helium*

gas as used by a chemist might refer to any [stable] mixture of helium isotopes, including pure ^{3}He. See Figure 6.

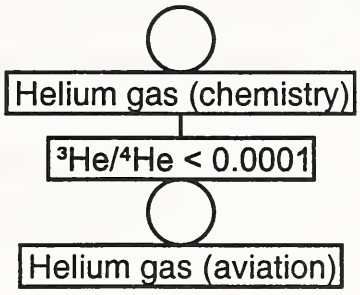

Figure 6: Concept lattice for helium gas

Any non-vacuous term can be made more specific, more general, or just different by adjusting the selection of essential properties. There is little grounds for optimism that different modellers would settle on the same ones. The ramifications for conceptual integrity are explored in detail in [27].

The conscious or unconscious act of selecting essential properties is a process of *abstraction* [28].

5 Identity

> In general, identity refers to the problem of being able to recognize individual entities in the world as being the same (or different).... Identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions). [1]
>
> Strictly speaking, identity is related to the problem of distinguishing a specific instance of a certain class from other instances by means of a *characteristic property*, which is unique for *it* (that *whole* instance). [2]

When some thing undergoes a change, whether or not it is considered the *same* thing afterwards depends on how that thing is identified.

Table 7 shows a many-valued formal context for the decay of the ^{5}He atom discussed in the previous section.

	Time	Protons	Neutrons
1	t_1	2	3
2	t_2	2	2

Table 7: Many-valued formal context for ^{5}He decay

Using nominal scales to transform the many-valued attributes, the formal context shown in Table 8 results.

As yet, no commitment has been made regarding whether the ^{4}He is the same atom as the ^{5}He. This formal context merely models observations of phenomena (1 and 2), which correspond to actual objects in Temporal Concept Analysis (see Definition 8).

	t_1	t_2	2 protons	2 neutrons	3 neutrons
1	×		×		×
2		×	×	×	

Table 8: Formal context for ^{5}He decay

The decision to identify the two observations with the same atom is modelled with a CTSOT whose formal context is shown in Table 9 and whose time relation is (He,1) → (He,2). The life track of He is reflected by the arrow in the corresponding concept lattice shown in Figure 7.

	t_1	t_2	2 protons	2 neutrons	3 neutrons
(He,1)	×		×		×
(He,2)		×	×	×	

Table 9: Formal context for ^{5}He decay with one atom

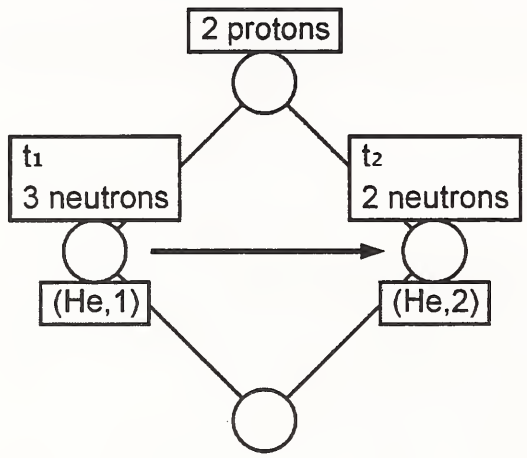

Figure 7: Concept lattice for ^{5}He decay with life track

If instead one chooses to view the ^{4}He as a different atom, one simply renames the actual objects (see Table 10). In this case the life tracks of the atoms disappear since there is only one applicable actual object for each of them. However, the formal context is structurally identical.

This example demonstrates that the decision to identify the two observations with the same atom or with different atoms is somewhat arbitrary. It is a subjective interpretation of objective phenomena. In contrast, some modelling environments treat identity as a haecceity [26] and cannot accurately represent the relationships among alternate identities.

As with essential properties, the conscious or unconscious act of selecting identity criteria is a process of abstraction.

That the appropriateness of any given selection of identity criteria is relative to the application is easily shown using a classic example.

> The ship on which Theseus sailed with the youths and returned in safety, the thirty-oared galley, was preserved by the Athenians down to the time of

	t_1	t_2	2 protons	2 neutrons	3 neutrons
(^{5}He,1)	×		×		×
(^{4}He,2)		×	×	×	

Table 10: Formal context for ^{5}He decay with two atoms

> Demetrius Phalereus. They took away the old timbers from time to time, and put new and sound ones in their places, so that the vessel became a standing illustration for the philosophers in the mooted question of growth, some declaring that it remained the same, others that it was not the same vessel. [29]

If the application of the ship that existed in the time of Demetrius Phalereus were a historical exhibit by which to remember the life of Theseus, then identity criteria loose enough to admit a patched and restored version of the ship that Theseus took to sea would suffice. To say "This is not the ship of Theseus" because some rotten wood was replaced would be misleading. But if the application were an investigation of the safety and seaworthiness of the vessel to determine whether Theseus was guilty of an act of negligence, then every board of the ship would need to be as it was when Theseus took it to sea. To say "This *is* the ship of Theseus" after work affecting its seaworthiness had been completed would be misleading.

6 Unity

> Unity refers to the problem of describing the way the parts of an object are bound together, such that we know in general what is part of the object, what is not, and under what conditions the object is a whole. [1]
>
> Unity... is related to the problem of distinguishing the *parts* of an instance from the rest of the world by means of a *unifying relation* that binds them together (not involving anything else). [2]

As noted by Melissus, unity relates to the more general notion of boundaries.

> If it were not one, it would form a boundary in relation to something else. [30]

One can define a thing by selecting spatial and temporal boundaries, deciding what is part of it and what is not. As with essential properties and identity criteria, the conscious or unconscious act of selecting boundaries is a process of abstraction. When formalized, the process is analogous to the examples in the previous section, where one subjectively picks objects out of a soup of observations.

It is often if not always the case that the spatial and/or temporal boundaries of a thing as people conceive of it are *vague* [31, 32]. Any precise model of such boundaries, including one using Formal Concept Analysis, necessarily adds arbitrary and application-specific constraints. However, one can use conceptual scaling to minimize the impact.

For example, once again considering the decay of ^{5}He, it is unlikely that the precise instant at which the atom decayed (or the instants at which the process of decay began and ended) would be known. However, it would be known with some certainty that it had not yet decayed at time t_1, and that it had already decayed at time t_2. With appropriate scaling, one need only consider those time granules for which precise knowledge is available.

Atomic decay makes an especially interesting example to illustrate the concept of unity because the parts of the whole do not, in fact, have identity. It is meaningless to ask *which* neutron was the escapee, or even whether the neutrons in the ^{4}He are the same ones as were in the ^{5}He. Any neutron is as good as any other. Nevertheless, the helium atom retains *its* identity.

Table 11 and Figure 8 model the ^{5}He decay at the subatomic level, with space and time scaled to match the spatial and temporal extents of the atoms identified in Section 5. He is effectively a coarse spatio-temporal granule that includes the finer granules ^{5}He and ^{4}He. He, ^{5}He and ^{4}He have identical spatial extents but different temporal extents. Which one would be intended by an unelaborated reference to "the atom" depends on the application.

The actual object (n3,2), representing the observation of a third neutron at time t_2, is within the temporal extents of He and ^{4}He but outside of their spatial extents.

The naming of the actual objects in this example is consistent with a notional identification of two protons and three neutrons having distinguishable life tracks. However, as stated above, this identification is meaningless. Fortunately, it is also superfluous in the formal context. One can rename the actual objects so that no connection between the particles observed at time t_1 and those observed at time t_2 is suggested, and the resulting context is structurally identical.

Using duality, one can reify the atoms as shown in Table 12.

	He	^{5}He	^{4}He
(p1,1)	×	×	
(p2,1)	×	×	
(n1,1)	×	×	
(n2,1)	×	×	
(n3,1)	×	×	
(p1,2)	×		×
(p2,2)	×		×
(n1,2)	×		×
(n2,2)	×		×
(n3,2)			

Table 11: Formal context for ^{5}He decay with notional subatomic particles

	(p1,1)	(p2,1)	(n1,1)	(n2,1)	(n3,1)	(p1,2)	(p2,2)	(n1,2)	(n2,2)	(n3,2)
He	×	×	×	×	×	×	×	×	×	
^{5}He	×	×	×	×	×					
^{4}He						×	×	×	×	

Table 12: Dual context for ^{5}He decay with notional subatomic particles

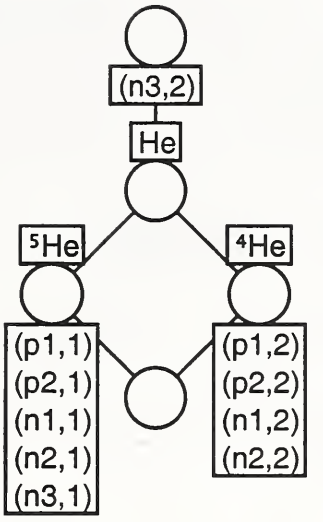

Figure 8: Concept lattice for ^{5}He decay with notional subatomic particles

7 Possibility

Possible things is a way of saying "things that might actually exist, but that we do not *know* exist;" alternately, "things that could potentially exist someday, but that do not exist now." Any hypothesized class that does not intend a logical contradiction could possibly have instances. For example, the class of ^{11}He atoms.

Let s be the statement "an ^{11}He atom actually exists." s is either true or false. Letting the truth attribute be represented by p, the falsehood attribute by $\sim p$, the concept lattice for Boolean logic is shown in Figure 9.

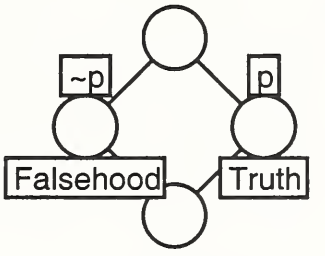

Figure 9: Concept lattice for Boolean logic

Modal logic introduces distinctions between statements that are *possibly* true ($\Diamond p$), those that are *materially* true (p), and those that are *necessarily* true ($\Box p$). If one wishes to model these distinctions, one can use the interordinal scale of truth values shown in Table 13. The corresponding lattice is shown in Figure 10.

The uncertainty about possible things is not logical, but epistemic in nature. One accepts that $s \vee \sim s$ is a tautology even if one does not know whether s or $\sim s$ are individually true. Nevertheless, one can model the intent by interpreting the attributes epistemically and naming the unnamed concepts.

Figure 11 shows the concept lattice for an epistemic modal logic. The names of the attributes have been prefixed by "know," reflecting the shift from properties to *known* properties (a

	$\sim\Diamond p$	$\sim p$	$\Diamond\sim p$	$\Diamond p$	p	$\Box p$
Contradiction	×	×	×			
Counterfactual		×	×	×		
Fact			×	×	×	
Tautology				×	×	×

Table 13: Formal context for modal logic

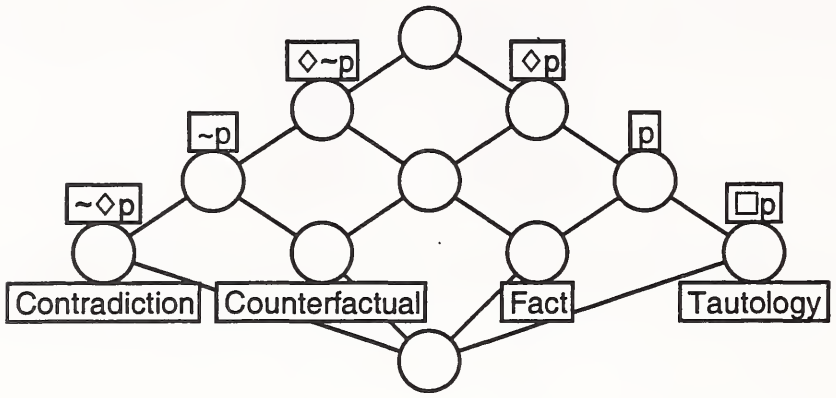

Figure 10: Concept lattice for modal logic

statement can be true without one knowing that it is true, but not vice-versa). The naming of concepts is admittedly arbitrary, but it is significant that there can be as many as ten distinct, meaningful modalities of knowledge. If the logical modalities are discarded, it reduces to epistemic Boolean logic, shown in Figure 12.

For one who is modelling with a context less rich than epistemic modal logic, the concept *unknown* is a valid generalization of *possibility* or *negative possibility*. However, neither *truth* nor *falsehood* would be a valid substitution for either one. Similarly, it would be an error to substitute *tautology* for either *truth* or *fact*, or to substitute *contradiction* for either *falsehood* or *counterfactual*. If, for example, the counterfactual "an ^{11}He atom actually exists" were made a contradiction (e.g., by declaring the class of ^{11}He atoms and the class *actual thing* to be disjoint), the model would be invalid if there were any time and place at which somebody managed to manufacture an ^{11}He atom. (One could argue that it is invalid *a priori* according to the formal semantics of necessity, but the debate is academic unless and until the counterexample appears.)

It is the tendency to make these invalid substitutions that creates the possible things reuse problem. A context that does not support *unknown* clearly leaves the modeller with little choice other than to make invalid substitutions. But too many *unknowns* results in a vacuous model—all things are possible; nothing can ever be ruled out. The treatment of possible things thus becomes a compromise between the desire for a generally valid model and the desire for a model that is constrained enough to enable the application for which it was built.

Subjectively, one might be less likely to declare the unreality of ^{11}He than of a more fanciful hypothetical, e.g., unicorns. Objectively, however, one can never disprove the existence of any hypothesized entity. One can argue relative likelihoods based on observations, but the step from unlikely to impossible requires a leap of faith (or rather the opposite—a leap of

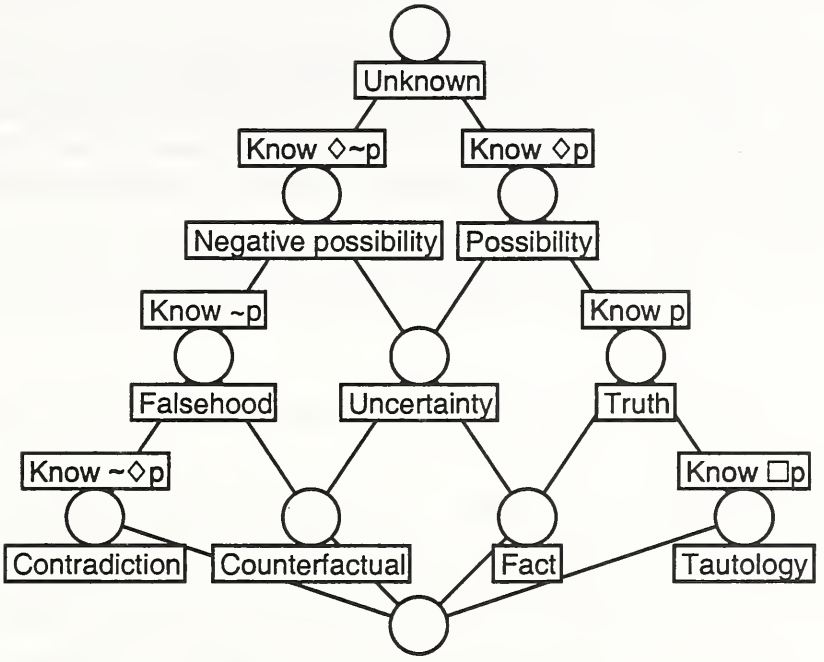

Figure 11: Concept lattice for epistemic modal logic

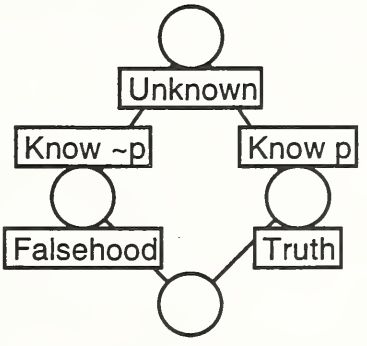

Figure 12: Concept lattice for epistemic Boolean logic

skepticism?).

8 Tense

Since it is the goal of modellers to capture knowledge of lasting value, the question of how to model the past and future as distinct from the present is often ignored. The resulting model is timeless in the sense of having no concept of time whatsoever. Things simply are as they are, unchanging; or, if things do change, the result is a *different* model. There is no formal connection between the old and the new.

Timeless models (e.g., OWL ontologies) essentially exist in the "long now" contemplated by Augustine.

> If there are times past and future, I desire to know where they are. [33]

Those modellers who do model time choose to structure it in different ways. Tense logic structures time in terms of past, present, and future (a.k.a. the A-series or the tenser approach) [34]. UML sequence diagrams structure time in terms of earlier and later (a.k.a. the B-series or the detenser approach). Process Specification Language (PSL) [35] structures time in terms of reified time points (a.k.a. the four-dimensional approach).

These different structures of time can be reconciled using conceptual scaling as exemplified in Figure 13. M1 is a tenser model whose present is between 08:00 and 10:00; M2 is a tenser model whose present is between 09:00 and 12:00. t_1 is an event occurring earlier than t_2 in a detenser model; one need not know exactly when they occurred, but the example reflects an assumption that they both occurred some time between 10:00 and 11:00. The timeless models M3 and M4 are assigned time granules corresponding to the times at which they are valid (from 09:00 to 11:00 for M3, always for M4).

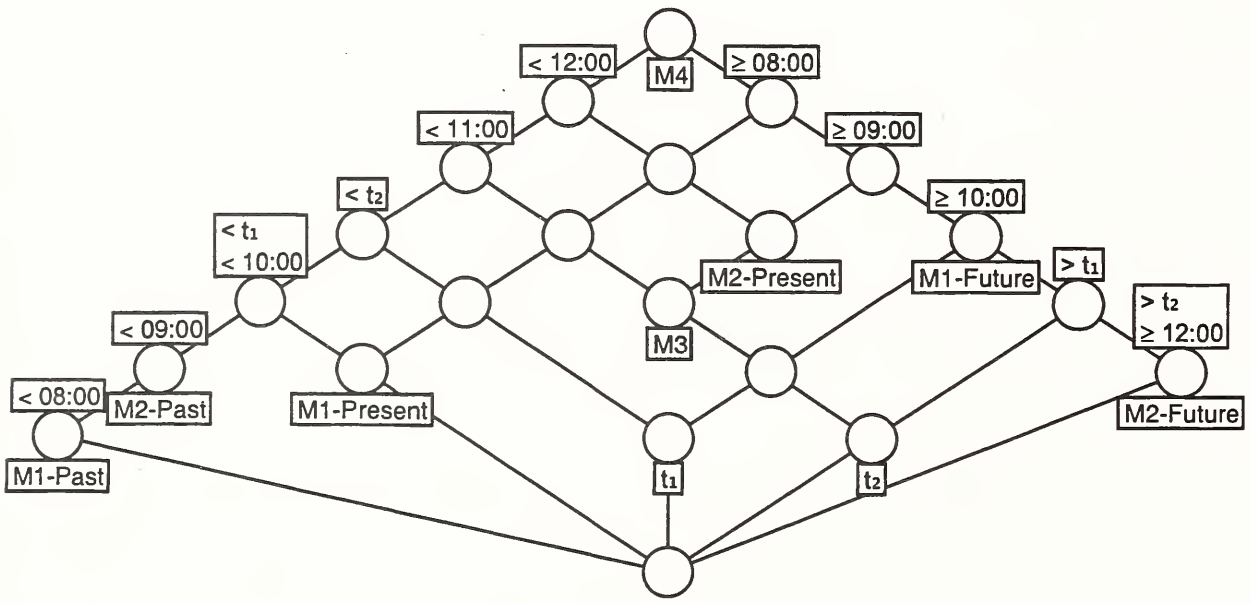

Figure 13: Integrated time scale

Any time scales can be integrated in this fashion provided that all of the temporal relations can be discovered. Of course, it is optimistic to assume that one would always be able to discern the intended time scales of different models with sufficient precision to get all of the temporal relations, let alone map them onto clock time as shown in Figure 13. Fewer temporal relations results in coarser time granules. One knows that a granule is *too* coarse if it appears that any object should have two different situations within that granule (a violation of the unique-state theorem [8, Section 3.3]).

The "long now" of a static model (one granule for all time, as with M4) would be sufficient if, in the application of interest, no contradictions would result from instantiating all things past, present, and future as if they were contemporaries. However, it would not be sufficient for any application that needed to deal with change, since any kind of change would mean that some object had two different situations within the granule.

9 Realism

> Absolute equality, absolute beauty, any absolute existence, true being—do they ever admit of any change whatsoever? ... It must necessarily remain the same.... But how about the many things, for example, men, or horses, or cloaks, or any other such things, which bear the same names as the absolute essences and are called beautiful or equal or the like? ... You can see these and touch them and perceive them by the other senses, whereas the things which are always the same can be grasped only by the reason, and are invisible and not to be seen. [36]

As was mentioned in Section 4, classifying things is a process of abstraction. But some reject the claim that classes are abstractions of a higher level than the things classified.

Some modelling architectures segregate levels of abstraction (*fixed architecture*); others do not (*flat architecture*). The desirability of strict separation is a topic of debate. There are intuitively attractive notions that cannot be rendered faithfully using a fixed architecture. For example, consider the class that is called *class*. Intuitively, *class* is an instance of itself. But proponents of fixed architecture argue that it is confused thinking to identify any instance of a class with the class itself, and that doing so produces a model that has no sensible interpretation by man or machine [28, 37], or at best an unconventional interpretation that does not integrate readily with conventional logic [38]. In any event, it certainly invalidates the set-theoretic interpretation of classification.

Ultimately, architectural support for separating levels of abstraction is only a convenience. It cannot enforce a way of thinking. A realist can always sabotage a fixed architecture by creating individuals that are interpreted as classes and relationships that are interpreted as instantiation. But in doing so, he or she abandons the formal semantics of that architecture and becomes dependent on a nonstandard, revisionist interpretation. In a similar way, one can interpret levels of abstraction onto a flat architecture if one is willing to replace that architecture's classification and inference rules with more restricted ones.

Table 14 highlights the concept called *possibility* (medium font, outlined) and the exemplary object with the same name (bold font, double outlined) from Figure 11. Clearly, they are

not identical. The extent of the formal concept ({Counterfactual, Uncertainty, Possibility, Truth, Fact, Tautology}, {Know $\Diamond p$}) includes all objects having the necessary attribute, but the exemplary object (Possibility) has no inherent relationship to those other objects (Counterfactual, etc.). The revisionist interpretation recognizes a special relationship between the exemplary object and the others that is unsupported by the formal semantics of the model.

	Know $\sim\Diamond p$	Know $\sim p$	Know $\Diamond\sim p$	**Know $\Diamond p$**	Know p	Know $\Box p$
Contradiction	×	×	×			
Counterfactual		×	×	×		
Falsehood		×	×			
Negative possibility			×			
Uncertainty			×	×		
Possibility				×		
Truth				×	×	
Fact			×	×	×	
Tautology				×	×	×
Unknown						

Table 14: The object Possibility and its object concept

The concept whose intent consists of all attributes of an object g and whose extent consists of all objects having those attributes is called the *object concept* of the object g and is denoted by γg [3, Definition 22]. Formally, g and γg are not even comparable; one is an object, the other is a concept.

In some applications, equivocation between g and γg might be safe if g were the only object in the *contingent* (the set of all objects belonging to the extent of the concept but not to the extent of any subconcept [39]) of γg. In other applications, it might be safe only if g were the only object in the extent of γg. But this equivocation is always formally incorrect and inherently risky. There will be many applications in which it is completely unsafe.

The decision to use a fixed architecture or not is a technical one, influenced by the relative expedience of expressing the concepts needed to serve particular applications. However, as the example shows, the impedance mismatch between fixed and flat architectures is significant.

10 Intent

> Mathematicians, generally, have an inclination toward extension, 'philosophers' toward intension. Now, it is interesting to note that mathematicians have a record of continuous constructive progress, and at each epoch have produced the highest kind of language known.... The 'philosophers', in the main, have a record of failure. [28, p. 176]

Debate over the relative value of intent and extent continues with considerable energy. Some who favor a flat modelling architecture argue that it is more amenable to intensional definition of classes (which, presumably, is a good thing). The following definitions and discussion, while intended to address logics in particular, nonetheless illustrate the dilemma faced by modellers.

> **Extensional** (adj., of a logic) A set-based theory or logic of classes, in which classes are considered to be sets, properties considered to be sets of <object, value> pairs, and so on. A theory which admits no distinction between entities with the same extension. See *Intensional.*
>
> **Intensional** (adj., of a logic) Not extensional. A logic which allows distinct entities with the same extension.
>
> (The merits and demerits of intensionality have been extensively debated in the philosophical logic literature. Extensional semantic theories are simpler, and conventional semantics for formal logics usually assume an extensional view, but conceptual analysis of ordinary language often suggests that intensional thinking is more natural. Examples often cited are that an extensional logic is obliged to treat all 'empty' extensions as identical, so must identify 'round square' with 'santa clause', and is unable to distinguish concepts that 'accidentally' have the same instances, such as human beings and bipedal hominids without body hair. The semantics described in this document is basically intensional.) [12, Glossary]

Formal Concept Analysis reduces this dichotomy to a mathematical extreme, defining formal concepts in such a way that intensional definitions (in terms of necessary and sufficient attributes) and extensional definitions (in terms of objects) entail one other via a formal mapping. However, it is possible to combine intent and extent in other ways, sacrificing the ability to map between intent and extent in order to gain other useful abilities. For example, reference [27] uses a definition of class that incorporates the extensional characteristic of taking class membership as primitive and the intensional characteristic of defining necessary (but not sufficient) conditions for class membership. This compromise allows one to associate essential properties with classes without presuming that the set of properties is sufficient to define the class with respect to all possible and future things.

In a static universe, extensional definitions are sufficient. If two concepts have the same extent, then they are interchangeable within the scope of the static universe and there is no value in distinguishing them. The value of intent is in making statements regarding possible and future individuals. Unfortunately, these statements become problematic when some possible individual that breaks the assumptions of the modeller becomes actual or becomes known. Whether or not it is necessary to make statements about possible and future individuals or to distinguish concepts having the same extents is clearly application-specific, as is the selection of essential properties for intensional definitions (see Section 4).

To illustrate, Figure 14 shows a conceptual model used by a hypothetical knowledge engineer who is building a knowledge base. The engineer defines *knowledge* intensionally as a collection of true statements. Truth1 is a true statement, e.g., "All men are mortal." p is the truth attribute again as in Section 7.

Figure 14: Original conceptualization

Figure 15 shows that conceptual model a while later. A new piece of knowledge has been added. Truth2 is also a true statement, e.g., "John Doe is a man."

Figure 15: Appearance of counterexample

At this point it occurs to the knowledge engineer that the intensional definition of knowledge as a collection of true statements was not what he or she intended after all. While the statement "John Doe is a man" is true, it is only *materially* true, not *necessarily* true like the statement "All men are mortal." John Doe could be run over by a bus and cease being a man at any time. So the engineer's conceptualization evolves, now defining *knowledge* as a collection of *necessarily* true statements.

Figure 16 shows an integration of the old and new conceptualizations. At time t_1, the attributes $\Diamond\sim p$ and $\Box p$ effectively do not exist, and Truth1 has only the attribute p. At time t_2, the attributes $\Diamond\sim p$ and $\Box p$ are created, making it possible to distinguish facts from tautologies. The new attribute $\Box p$ is ascribed to Truth1, effectively narrowing its classification.

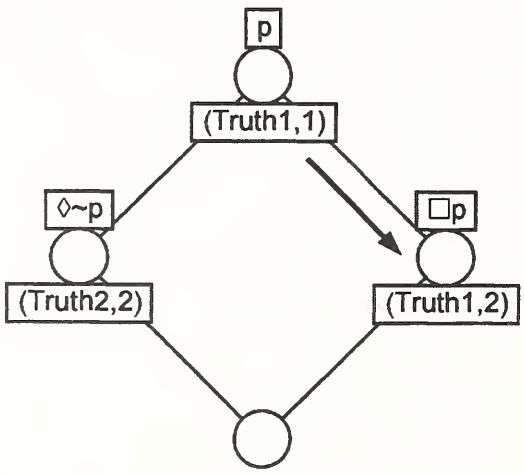

Figure 16: Evolution of conceptualization

With a purely extensional approach, facts and tautologies collapse into the same concept at time t_1, but are distinguishable at time t_2, after the counterexample appears. The intent, as such, manifests indirectly in the engineer's decisions to place certain individuals in certain sets (the observable action resulting from an unobservable mental event) but is

never explicitly stated. With a purely intensional approach, the intent is made explicit at time t_1, but comes back to haunt at time t_2, when the modeller is obliged either to classify something incorrectly or to make incompatible changes to the model. With the approach in [27], an intent is made explicit at time t_1, but the modeller is not obliged to classify something incorrectly or invalidate the model at time t_2.

11 Compatibility for integration

When concepts from different applications are formalized and integrated into a single concept lattice, the ramifications of substituting a concept supported by one application for a concept supported by a different application, for a given set of objects, are immediately apparent. If the two concepts collapse into one in the concept lattice, then substitution is valid. Generalization—the substitution of a superconcept for one of its subconcepts—is also valid: if a given object has the attributes to be in the extent of the subconcept, then it necessarily has the subset of those attributes to be in the extent of the superconcept. All other substitutions are invalid.

In cases where it is believed that a generally invalid substitution would be valid under certain conditions, that validity can be proven or disproven by constructing a subcontext in which objects and/or attributes that violate those conditions are omitted. For example, returning to Figure 11, it is invalid to substitute *fact* for *truth* or *counterfactual* for *falsehood*. However, if one assumes that it will never happen that one knows $\Diamond p$ or $\Diamond{\sim}p$ without first knowing p or ${\sim}p$, respectively, then the attributes corresponding to knowledge of $\Diamond p$ and $\Diamond{\sim}p$ are redundant and can be deleted. By deleting them, one creates the concept lattice shown in Figure 17. The distinctions between *fact* and *truth* and between *counterfactual* and *falsehood* have disappeared, so the previously invalid substitutions are valid in the restricted subcontext.

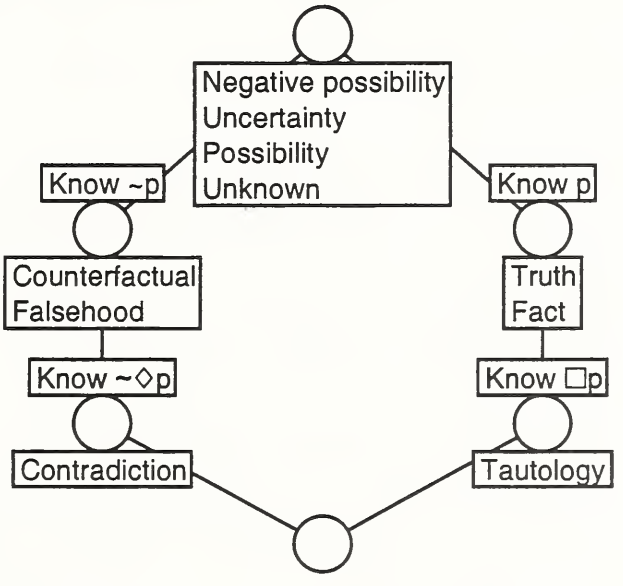

Figure 17: Concept lattice for reduced epistemic modal logic

Unfortunately, some models constructed under flat architectures cannot be faithfully reproduced using Formal Concept Analysis, which exhibits a fixed architecture, so a conclusive

analysis is not possible in every case.

A more thorough discussion of applying Formal Concept Analysis to integration problems can be found in [40].

12 Conclusion

This paper explored seven facets of relativity in explicit conceptual models using Formal Concept Analysis, demonstrating first that the capture of application-specific constraints is inextricable from the modelling process, and second that the semantic differences between models built for different applications can themselves be modelled formally in most cases. By analyzing those differences, one can determine whether the applications themselves are sufficiently compatible at the conceptual level to enable integration.

It should be noted that all formal models require validation to convince the principals that they accurately reflect reality. An analysis of an invalid model may be conclusive, but the conclusions are still invalid. This observation applies to all models that aspire to be more than detached mathematical constructs having nothing to do with anything. As such, a detailed exploration of validation concerns is beyond the scope of this paper.

13 Acknowledgments

The author thanks all those whose reviews and suggestions have improved this paper, including Edward Barkmeyer, Peter Becker, Joachim Hereth Correia, Steven Fenves, and Michael Grüninger.

The concept lattices were produced using ToscanaJ version 1.5 [41, 42, 43].

References

[1] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, February 2002.

[2] Nicola Guarino and Christopher Welty. Identity and subsumption. In Rebecca Green, Carol A. Bean, and Sung Hyon Myaeng, editors, *The Semantics of Relationships: An Interdisciplinary Perspective*, pages 111–125. Kluwer, 2002.

[3] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, Berlin, Heidelberg, and New York, 1999.

[4] Bernhard Ganter and Rudolf Wille. Conceptual scaling. In Fred Roberts, editor, *Applications of combinatorics and graph theory to the biological and social sciences*, pages 139–167, New York, 1989. Springer.

[5] Karl Erich Wolff. Concepts, states, and systems. In *Proceedings of the Third International Conference on Computing Anticipatory Systems (CASYS'99)*, pages 83–97, 1999.

[6] Karl Erich Wolff. Temporal concept analysis. In *Proceedings of the ICCS-2001 International Workshop on Concept Lattice-Based Theory, Methods, and Tools for Knowledge Discovery in Databases*, pages 91–107, Palo Alto, CA, 2001. Stanford University.

[7] Karl Erich Wolff. Interpretation of automata in temporal concept analysis. *Lecture Notes in Artificial Intelligence*, 2393:341–353, 2002. Alias, Integration and Interfaces, 10th International Conference on Conceptual Structures (ICCS 2002).

[8] Karl Erich Wolff and Wendsomde Yameogo. Time dimension, objects, and life tracks - a conceptual analysis. *Lecture Notes in Artificial Intelligence*, 2746:188–200, July 2003. Alias, Conceptual Structures for Knowledge Creation and Communication, 11th International Conference on Conceptual Structures (ICCS 2003).

[9] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.

[10] OMG Unified Modeling Language specification, version 1.5. OMG document formal/2003-03-01, Object Management Group, http://www.omg.org/cgi-bin/doc?formal/2003-03-01, March 2003.

[11] Graham Klyne and Jeremy J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax.* W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, February 2004.

[12] Patrick Hayes, editor. *RDF Semantics.* W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-mt-20040210/, February 2004.

[13] Dan Brickley and R. V. Guha, editors. *RDF Vocabulary Description Language 1.0: RDF Schema.* W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/, February 2004.

[14] Deborah L. McGuinness and Frank van Harmelen, editors. *OWL Web Ontology Language Overview.* W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-features-20040210/, February 2004.

[15] Michael K. Smith, Chris Welty, and Deborah L. McGuinness, editors. *OWL Web Ontology Language Guide.* W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-guide-20040210/, February 2004.

[16] Mike Dean and Guus Schreiber, editors. *OWL Web Ontology Language Reference.* W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-ref-20040210/, February 2004.

[17] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[18] Terry Halpin. Object Role Modeling (ORM/NIAM). In Peter Bernus, Kai Mertins, and Günter Schmidt, editors, *Handbook on Architectures of Information Systems*, chapter 4. Springer, 1998. Available from http://www.orm.net/.

[19] Terry Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design.* Morgan Kaufmann, 2001.

[20] EXPRESS language reference manual. ISO document 10303-11:1994, ISO, 1994.

[21] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, November 1995.

[22] Opencyc selected vocabulary and upper ontology, December 2002. http://www.cyc.com/cycdoc/vocab/vocab-toc.html.

[23] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pages 2–9, October 2001.

[24] SUMO ontology, 2004. Available at http://ontology.teknowledge.com/.

[25] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery*, 42(4):741–843, 1995.

[26] Richard Cross. Medieval theories of haecceity. In Edward N. Zalta, editor, *Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2003 edition, December 2003. Available at http://plato.stanford.edu/archives/fall2003/entries/medieval-haecceity/.

[27] David Flater. A logical model of conceptual integrity in data integration. *Journal of Research of the National Institute of Standards and Technology*, 108(5):395–402, September-October 2003.

[28] Alfred Korzybski. *Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics*. Institute of General Semantics, Englewood, New Jersey, 5th edition, 1994.

[29] Lucius Mestrius Plutarchus. Theseus, XXIII.1 (trans. Bernadotte Perrin). In *Lives*. Harvard University Press, 1914. Available from the Perseus Digital Library, http://www.perseus.tufts.edu/cgi-bin/ptext?lookup=Plut.+Thes.+23.1.

[30] Melissus of Samos. Fragment (trans. Greek to German, Hermann Diels; German to English, Kathleen Freeman). In Walter Kaufmann, editor, *Philosophic Classics, Volume I: Thales to Ockham*, page 32. Prentice-Hall, Englewood Cliffs, New Jersey, 2nd edition, 1968.

[31] Timothy Williamson. *Vagueness*. Routledge, 1996.

[32] Rosanna Keefe and Peter Smith, editors. *Vagueness: A Reader*. MIT Press, 1999.

[33] Augustine of Hippo. Confessions, book XI, chapter XVIII (trans. J. G. Pilkington). In Walter Kaufmann, editor, *Philosophic Classics, Volume I: Thales to Ockham*, page 513. Prentice-Hall, Englewood Cliffs, New Jersey, 2nd edition, 1968.

[34] Antony Galton. Temporal logic. In Edward N. Zalta, editor, *Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2003 edition, December 2003. Available at http://plato.stanford.edu/archives/win2003/entries/logic-temporal/.

[35] PSL Core Ontology, 2004. Available at http://www.mel.nist.gov/psl/psl-ontology/pslcore_page.html.

[36] Plato. Phaedo, 78d-79a (trans. Harold North Fowler). In *Plato in Twelve Volumes*, volume 1. Harvard University Press, 1966. Available from the Perseus Digital Library, http://www.perseus.tufts.edu/cgi-bin/ptext?lookup=Plat.+Phaedo+78d.

[37] Jeff Z. Pan and Ian Horrocks. Metamodeling architecture of web ontology languages. In *Proceedings of the Semantic Web Working Symposium*, pages 131–149, July 2001.

[38] Jeff Z. Pan and Ian Horrocks. RDFS(FA) and RDF MT: Two semantics for RDFS. *Lecture Notes in Computer Science*, 2870:30–46, October 2003. Alias, Proceedings of the Second International Semantic Web Conference.

[39] Joachim Hereth Correia, Gerd Stumme, Rudolf Wille, and Uta Wille. Conceptual knowledge discovery—a human-centered approach. *Applied Artificial Intelligence*, 17:281–302, 2003.

[40] Gerd Stumme and Alexander Mädche. FCA-merge: Bottom-up merging of ontologies. In Bernhard Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 225–230, Seattle, Washington, 2001.

[41] ToscanaJ, version 1.5, May 2004. Available from Sourceforge, http://toscanaj.sourceforge.net/.

[42] Peter Becker, Joachim Hereth, and Gerd Stumme. ToscanaJ – an open source tool for qualitative data analysis. In *Proceedings of the 2002 International Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases (FCAKDD 2002)*, pages 1–2, Lyon, France, July 2002.

[43] Peter Becker and Joachim Hereth Correia. The ToscanaJ suite for implementing conceptual information systems. In *Proceedings of the First International Conference on Formal Concept Analysis (ICFCA 03)*, 2003. To appear.

[44] Ontology Definition Metamodel RFP. OMG document ad/2003-03-40, Object Management Group, http://www.omg.org/cgi-bin/doc?ad/2003-03-40, March 2003.

Appendix

This appendix gives technical notes supporting the characterizations in Figure 4.

FCA

Objects have attributes. Two objects with the same attributes are distinguishable, so objects have haecceity.

Formal concepts have extent and intent. Two formal concepts with the same extent and intent are identical, so formal concepts do not have haecceity.

Conceptual Graphs

Individuals have haecceity.

> There is a set $I = \{\#1, \#2, \#3, \ldots\}$ whose elements are called *individual markers*. ... An individual marker is a surrogate for some individual in the real world, a perceived world, or a hypothetical world. [9, Section 3.3.1]
>
> This concept [[PERSON:Judy]] represents the indefinite form *a person named Judy*, but [PERSON:Judy#3074] represents *the person named Judy*. [9, Section 3.3.4]

Types have extent and intent. Two types can be the same, so they do not have haecceity.

> Whether two concept types are the same depends on their links to the semantic network rather than their external instances. [9, Section 3.2.1]

Attributes are represented using relations. Concepts appearing in a CG cannot represent types directly. The relations that types have to one another are at the meta level.

> The type hierarchy is a *higher-order relation* [9, Section 3.2.1]

UML

The characterization is of UML version 1.5 [10]. As of 2004-07-23, UML version 1.5 remains current, though UML 2.0 is nearing completion. The Ontology Definition Metamodel [44] is still in early stages with four initial submissions.

UML objects have inherent identity, hence haecceity.

> An object represents a particular instance of a class. It has identity and attribute values. [10, Section 3.39.1]

Composite objects are objects with parts.

> A composite object represents a high-level object made of tightly-bound parts. This is an instance of a composite class, which implies the composition aggregation between the class and its parts. [10, Section 3.40.1]

Classes have intent and extent.

> A *class* is the descriptor for a set of objects with similar structure, behavior, and relationships. The model is concerned with describing the intension of the class, that is, the rules that define it. The run-time execution provides its extension, that is, its instances. [10, Section 3.22]

Classes do not have attributes except at the meta level. In UML terminology, *attribute* refers to the "slot" attached to a class to indicate that every *instance* of that class—not the class itself—has a corresponding *attribute value.*

> An attribute is semantically equivalent to a composition association [10, Section 3.25.1]
>
> An association declares a connection (link) between instances of the associated classifiers (e.g., classes). [10, Section 2.5.4.1]

It is not possible in UML for two classes to be the same (if it were, they would not have a mapping into programming languages like C++), so UML classes have haecceity.

RDFS

Everything is a resource.

> All things described by RDF are called *resources*, and are instances of the class `rdfs:Resource`. [13, Section 2.1]

Resources have properties.

> The RDF Concepts and Abstract Syntax specification [11] describes the concept of an RDF property as a relation between subject resources and object resources. [13, Section 3]

Containers have members and properties.

> RDF containers are resources that are used to represent collections. ... Just as a hen house may have the property that it is made of wood, that does not mean that all the hens it contains are made of wood, a property of a container is not necessarily a property of all of its members. [13, Section 5.1]

Classes (as well as containers) are resources, so they also have properties.

By analogy with logical predicates, classes have intent but not haecceity.

> **Class** (n.) A general concept, category, or classification. Something used primarily to classify or categorize other things. Formally, in RDF, a resource of type `rdfs:Class` with an associated set of resources all of which have the class as a value of the `rdf:type` property. Classes are often called 'predicates' in the formal logical literature. [12, Glossary]

In general, there is nothing to prevent two resources from being the same and no assumption that they are distinct, so containers do not have haecceity either.

OWL

Individuals have properties.

> Properties can be used to state relationships between individuals or from individuals to data values. [14, Section 3.1]

Individuals can be the same, hence they do not have haecceity.

> *sameAs*: Two individuals may be stated to be the same. [14, Section 3.2]

OWL has sublanguages OWL Lite (not characterized), OWL DL and OWL Full. OWL Full classes can be individuals, and hence have properties; OWL DL classes cannot.

> OWL Full and OWL DL support the same set of OWL language constructs. Their difference lies in restrictions on the use of some of those features and on the use of RDF features. OWL Full allows free mixing of OWL with RDF Schema and, like RDF Schema, does not enforce a strict separation of classes, properties, individuals and data values. OWL DL puts constraints on the mixing with RDF and requires disjointness of classes, properties, individuals and data values. [16, Section 1.2]

Like RDFS classes, OWL classes do not have haecceity, although the sameness of two classes is only expressible in OWL Full (by treating them as individuals, with `sameAs`). The fact that two classes have the same extent but possibly different intent can be expressed in OWL DL with `equivalentClass`.

> In its simplest form, an equivalentClass axiom states the equivalence (in terms of their class extension) of two named classes. ... NOTE: The use of `owl:equivalentClass` does not imply class equality. Class equality means that the classes have the same intensional meaning (denote the same concept). ... Real class equality can only be expressed with the `owl:sameAs` construct. As this requires treating classes as individuals, class equality can only be expressed in OWL Full. [16, Section 3.2.2]

E-R

Entities have haecceity.

> An *entity* is a "thing" which can be distinctly identified. [17, Section 2.2]

Entities have *attributes* and *relationships* to other entities, both of which suffice as "attributes / properties" for the purpose of Figure 4.

> A *relationship* is an association among entities.
>
> An *attribute* can be formally defined as a function which maps from an entity set or a relationship set into a value set or a Cartesian product of value sets.... It maps a given entity to a single value (or a single tuple of values in the case of a Cartesian product of value sets). [17, Section 2.2]

Entity sets and entities are distinct, so entity sets do not have attributes.

Entity sets have extent and intent.

> Entities are classified into different *entity sets*.... There is a predicate associated with each entity set to test whether an entity belongs to it. [17, Section 2.2.1]

By analogy with logical predicates, entity sets do not have haecceity. If two entity sets have equivalent predicates, they are essentially identical.

ORM

Entities have haecceity.

> As a quality check at Step 1, we ensure that objects are well identified. ... *Entities* are "real world" objects that are identified by a definite description (e.g., the Academic with empnr 715). [18, Section 1.4]

A distinction is made between primitive (top-level) entity types and subtypes thereof with respect to haecceity. Primitive entity types have haecceity; subtypes do not.

> Each entity is an instance of a particular entity type (e.g., Person, Subject). [19, Section 3.3]
>
> It is possible that subtypes of a given entity type may overlap. However, *primitive entity types never overlap.* [19, Section 3.5]
>
> In general, B is a **proper subtype** of A if and only if $\text{pop}(B)$ is always a subset of $\text{pop}(A)$, and $A \neq B$. In this case, A is a *proper supertype* of B. If $A = B$, we do not specify any subtype connection. [19, Section 6.5]

Relationships are used instead of attributes. Relationships are between *objects* (entities or values), not types; entity types do not have attributes.

EXPRESS

Entities have extent (instances) and intent (common properties). Entity (data type) instances have values for the attributes that were declared on the entity (analogous to UML attributes on classes).

> **entity:** a class of information defined by common properties. [20, Clause 3.2.5]
>
> **entity data type:** a representation of an entity. An entity data type establishes a domain of values defined by common attributes and constraints. [20, Clause 3.2.6]
>
> **entity (data type) instance:** a named unit of data which represents a unit of information within the class defined by an entity. It is a member of the domain established by an entity data type. [20, Clause 3.2.7]

EXPRESS distinguishes *instance equality* (:=:) from *equality by value comparison* (=). Entity (data type) instances can be instance not equal even if they are equal by value comparison, so entity (data type) instances have haecceity.

> `a :=: b` evaluates to TRUE if `a` evaluates to the same entity instance as `b`, i.e., the implementation dependent identifiers are the same. [20, Clause 12.2.2.2]

Sets containing the same elements are instance equal, so sets do not have haecceity.

> Two sets `a` and `b` are instance equal if and only if each element in `a` is IN `b` and each element in `b` is also IN `a` [20, Clause 12.2.2.1]

There is no provision for the possibility that two entities would be the same, so entities have haecceity.

Cyc

Mathematical sets and collections do not have attributes (in the sense "spatial and temporal properties").

> All instances of #$Collection and all instances of #$Set-Mathematical (and thus all instances of #$SetOrCollection) are abstract entities, lacking spatial and temporal properties. [22, Mathematics, #$SetOrCollection]

Collections have extent and intent. Mathematical sets have only extent.

> Cyc collections are natural kinds or classes, as opposed to mathematical sets; their instances have some common attribute(s). Each Cyc collection is like a set in so far as it may have elements, subsets, and supersets, and may not have parts or spatial or temporal properties. Sets, however, differ from collections in that a mathematical set may be an arbitrary set of things which have nothing in common (see #$Set-Mathematical). In contrast, the instances of a collection will all have in common some feature(s), some 'intensional' qualities. In addition, two instances of #$Collection can be co-extensional (i.e., have all the same instances) without being identical, whereas if two arbitrary sets had the same elements, they would be considered equal. [22, Fundamental, #$Collection]

Collections are individuated by their intensional criteria for membership [22, Mathematics, #$SetOrCollection]. It follows that if the membership criteria of two collections are equivalent, they are indistinguishable; hence they do not have haecceity.

The ability to have spatial and temporal properties is divided in fine gradations among many subclasses of #$Individual. For brevity, attributes are simply ascribed to Cyc individuals in Figure 4.

It is possible to assert that two individuals are the same; therefore, individuals do not have haecceity.

> (#$equals THING1 THING2) means that THING1 and THING2 are numerically (as opposed to qualitatively) identical, i.e. they are one and the same thing. A sentence of the above form is true if and only if the terms occupying the two argument-places of '#$equals' denote the same thing. [22, Fundamentals, #$equals]

Groups are composite objects with spatial and temporal properties. Like other individuals, they lack haecceity.

> Each instance of #$Group is a composite object made up of one or more individual objects or events. A group is related to each of its members by the predicate #$groupMembers (q.v.). Note that instances of #$Group are *not* collections. A group has temporal extent and might have spatial location, while a collection is timeless and nonspatial. [22, Groups, #$Group]

SUMO

SUMO objects have spatial and temporal properties.

> Corresponds roughly to the class of ordinary objects. Examples include normal physical objects, geographical regions, and locations of Processes, the complement of Objects in the Physical class. In a 4D ontology, an Object is something whose spatiotemporal extent is thought of as dividing into spatial parts roughly parallel to the time-axis. [24, Object]

Classes have extent and intent; sets have only extent.

> Classes are not assumed to be extensional. That is, distinct Classes might well have exactly the same instances. ... Classes typically have an associated 'condition' that determines the instances of the Class. [24, Class]
>
> Sets are extensional—two Sets with the same elements are identical. ... A Set can be an arbitrary stock of objects. That is, there is no requirement that Sets have an associated condition that determines their membership. [24, Set]

Collections have spatial and temporal properties.

> Collections have members like Classes, but, unlike Classes, they have a position in space-time and members can be added and subtracted without thereby changing the identity of the Collection. Some examples are toolkits, football teams, and flocks of sheep. [24, Collection]

Corpuscular objects have parts.

> A SelfConnectedObject whose parts have properties that are not shared by the whole. [24, CorpuscularObject]

Nothing has haecceity. It is possible to assert that any two entities (objects, classes, etc.) are the same.

> (equal ?ENTITY1 ?ENTITY2) is true just in case ?ENTITY1 is identical with ?ENTITY2. [24, equal]

F-logic

Frame Logic makes no formal distinction between classes and individual objects. Any F-logic object can play the role of a class in subclassing and membership relationships. The word "class" is used informally to describe objects that so participate. Any object can have properties.

Objects (including classes) do not have haecceity. It is possible for two objects to be the same.

> Equality may be derivable even if it is not mentioned explicitly. For instance, $\{a :: b, b :: a\} \models a = b$ and $\{a[attr \rightarrow b], a[attr \rightarrow c]\} \models b = c$. [25, Appendix A]